CS 171 - Cryptography

Sanjam Garg

Lecture 24

Take Away from this Class

Definitions

Definitions

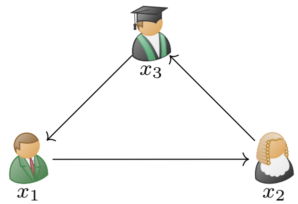
Definitions

Plan for today

- Multiparty Secure Computation
- Review of Definitions

Multiparty Secure Computation

- Parties P_1, P_2, P_3 hold private inputs $x_1, x_2, x_3 \in \{0,1\}^\ell$.
- Want to jointly compute a public circuit $C : (\{0,1\}^\ell)^3 \to \{0,1\}$ on their private inputs.
- Want to disclose only the output of the compuation.
- Are allowed to interact (and sample random coins).

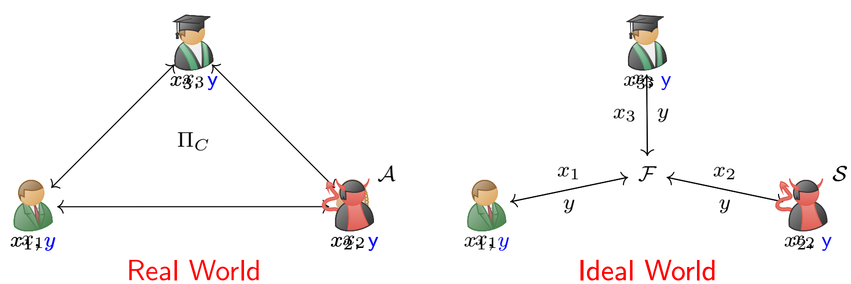


- Assume private and authenticated channels between every pair of parties.

Application

- Private contact discovery
- Bitcoin Wallets - Threshold Signing for ECDSA

Multiparty Secure Computation — Definition



- The adversary can be **malicious** or **honest but curious**.
- Correctness: $y = C(x_1, x_2, x_3)$.
- Security Informally: whatever $\mathcal{A}$ learns in the real world could be learnt in ideal world as well!
- Security: $\forall$ $\mathcal{A}$ there exists $\mathcal{S}$ such that no machine can distinguish between $REAL_{\Pi,A}(x_1, x_2, x_3)$ and $IDEAL_{\mathcal{F},\mathcal{S}}(x_1, x_2, x_3)$.

$(2,3)-$Threshold Secret Sharing

- Let $s \in \{0,1\}^m$. How do we $(2,3)$-secret share s?
- Share(s) : Sample $r_1, r_2 \leftarrow \{0,1\}^m$. Set $r_3 = s \oplus r_1 \oplus r_2$ and output $s_1 = (r_1, r_2), s_2 = (r_2, r_3)$ and $s_3 = (r_3, r_1)$.
- Reconstruct(s_i, s_j): Outputs $r_1 \oplus r_2 \oplus r_3$ where r_1, r_2, r_3 can be recovered from s_i, s_j.

MPC Protocol — Invariant and Input Secret Sharing

- Parties want to compute circuit C with $\oplus$ and $\times$ gates.
- **Invariant:** Parties compute a $(2,3)$ secret-sharing for each wire in the circuit.
- **Input Secret Sharing:** P_1, P_2, P_3 hold x_1, x_2, x_3 respectively. How do they recieve a $(2,3)$ secret sharing of these inputs?
- P_1 generates a $(2,3)$-secret sharing of its input x_1, keeps one share locally and passes the other two shares to P_2 and P_3. P_2 and P_3 do the same with their inputs.

MPC Protocol — $\oplus$ **Gate**

- P_1, P_2, P_3 hold (r_1, r_2), (r_2, r_3) and (r_3, r_1) such that $r_1 \oplus r_2 \oplus r_3 = \alpha$ and (s_1, s_2), (s_2, s_3) and (s_3, s_1) such that $s_1 \oplus s_2 \oplus s_3 = \beta$. How can parties compute a $(2, 3)$ secret sharing of $\alpha \oplus \beta$?
- Observe $\alpha \oplus \beta = (r_1 \oplus s_1) \oplus (r_2 \oplus s_2) \oplus (r_3 \oplus s_3)$. Thus, parties can set $(r_1 \oplus s_1, r_2 \oplus s_2)$, $(r_2 \oplus s_2, r_3 \oplus s_3)$ and $(r_3 \oplus s_3, r_1 \oplus s_1)$ as their $(2, 3)$ shares of $\alpha \oplus \beta$.

MPC Protocol — $\times$ **Gate**

- P_1, P_2, P_3 hold (r_1, r_2), (r_2, r_3) and (r_3, r_1) such that $r_1 \oplus r_2 \oplus r_3 = \alpha$ and (s_1, s_2), (s_2, s_3) and (s_3, s_1) such that $s_1 \oplus s_2 \oplus s_3 = \beta$. How can parties compute a $(2, 3)$ secret sharing of $\alpha \times \beta$?
- $\alpha \times \beta = (r_1 \oplus r_2 \oplus r_3) \cdot (s_1 \oplus s_2 \oplus s_3)$
  $= r_1 \cdot s_1 \oplus r_1 \cdot s_2 \oplus r_2 \cdot s_1$
  $\oplus\ r_2 \cdot s_2 \oplus r_2 \cdot s_3 \oplus r_3 \cdot s_2$
  $\oplus\ r_3 \cdot s_3 \oplus r_3 \cdot s_1 \oplus r_1 \cdot s_3$.
- P_1, P_2, P_3 can locally compute $t_1 = r_1 \cdot s_1 \oplus r_1 \cdot s_2 \oplus r_2 \cdot s_1$, $t_2 = r_2 \cdot s_2 \oplus r_2 \cdot s_3 \oplus r_3 \cdot s_2$ and $t_3 = r_3 \cdot s_3 \oplus r_3 \cdot s_1 \oplus r_1 \cdot s_3$ respectively.
- This is a $(3, 3)$ secret sharing. How do we go back to a $(2, 3)$ secret sharing?
- P_1 just sends its share with P_2 and so on!
- Also, rerandomize before sharing. P_i updates its share from t_i to $t_i \oplus u_i$ before sharing. Where u_1, u_2, u_3 are random shares such that $u_1 \oplus u_2 \oplus u_3 = 0$.

MPC Protocol — Output Reconstruction

- How do paties reconstruct the output given that they hold a $(2,3)-$secret sharing of the output wire?
- Each party publishes its shares and output can reconstructed.

Perfect Security

eav is for Eavesdropper

$\mathrm{PrivK}^{\mathrm{eav}}_{\mathrm{A},\Pi}$

1. A outputs $m_0, m_1 \in \mathcal{M}$.
2. $\mathrm{b} \leftarrow \{0,1\}$, $k \leftarrow$ Gen(), $c^* \leftarrow Enc_k(m_b)$
3. c^* is given to A
4. A output b'
5. Output 1 if $b = b'$ and 0 otherwise

Challenge ciphertext

Encryption scheme $\Pi = (Gen, Enc, Dec)$ with message space $\mathcal{M}$

is perfectly indistinguishable if

$\forall$ A it holds that:

$$\Pr\left[\mathrm{PrivK}^{\mathrm{eav}}_{\mathrm{A},\Pi} = 1\right] = \frac{1}{2}$$

A can always succeed with probability ½. How?

Drawback: Large Keys

CPA-Security

$\mathrm{PrivK}^{\mathrm{CPA}}_{\mathrm{A},\Pi}(n)$

1. Sample $k \leftarrow$ Gen(1^n), $A^{Enc_k(\cdot)}$ outputs $m_0, m_1 \in \{0,1\}^*, |m_0| = |m_1|$.
2. $\mathrm{b} \leftarrow \{0,1\}$, $c^* \leftarrow Enc_k(m_b)$
3. c^* is given to $A^{Enc_k(\cdot)}$
4. $A^{Enc_k(\cdot)}$ output b'
5. Output 1 if $b = b'$ and 0 otherwise

Encryption scheme $\Pi = (Gen, Enc, Dec)$ has indistinguishable encryptions under chosen-plaintext attack, or is *CPA-secure* if

$\forall$ PPT A it holds that:

$$\Pr\left[\mathrm{PrivK}^{\mathrm{CPA}}_{\mathrm{A},\Pi} = 1\right] \leq \frac{1}{2} + \mathrm{negl(n)}$$

Only PPT attackers and allowed some failure probability.

EAV Security

$\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\Pi}(n)$

1. $(pk, sk) \leftarrow G(1^n)$ and give pk to A.
2. A outputs $m_0, m_1 \in \{0,1\}^*, |m_0| = |m_1|$.
3. b ← {0,1}, $c \leftarrow Enc(pk, m_b)$
4. c is given to A and it outputs b'
5. Output 1 if $b = b'$ and 0 otherwise

Encryption scheme $\Pi = (Gen, Enc, Dec)$ is indistinguishable in the presence of an eavesdropper, or is *EAV-secure* if

∀ PPT A it holds that:

$$\Pr\left[\mathrm{PubK}^{\mathrm{eav}}_{\mathrm{A},\Pi} = 1\right] \leq \frac{1}{2} + \mathrm{negl(n)}$$

Pseudorandom Function (PRF)

Let $F\colon \{0,1\}^* \times \{0,1\}^* \rightarrow \{0,1\}^*$ be an efficient, length-preserving, keyed function. F is a PRF if for all PPT distinguishers D, there is a negligible function $negl(\cdot)$ such that:

$$\left|\Pr\left[D^{F_k(\cdot)}(1^n) = 1\right] - \Pr\left[D^{f(\cdot)}(1^n) = 1\right]\right| \leq negl(n)$$

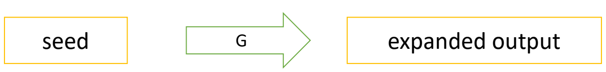
where $k \leftarrow U_n$ and $f \leftarrow Func_n$.

One-Way Functions: Formally

- A function $f: \{0,1\}^* \to \{0,1\}^*$ is a one-way function if:
- (easy to compute) There exists a polynomial-time algorithm M_f computing f; i.e., for all x, $M_f(\mathrm{x}) = f(x)$.
- (hard to invert) For all PPT A, there is a negligible function $negl$ such that
$$\Pr_{x \leftarrow \{0,1\}^n}\left[A\left(1^n, f(x)\right) \in f^{-1}\left(f(x)\right)\right] \leq negl(n)$$

Pseudorandom Generators

- $G: \{0,1\}^n \to \{0,1\}^{\ell(n)}$, where $\ell(n) > n$



- G is pseudorandom generator if $\forall$ PPT A we have $\exists$ $negl(\cdot)$ such that,
$$\left|\Pr_{x \leftarrow U_{\ell(n)}}[A(x) = 1] - \Pr_{s \leftarrow U_n}[A(G(s)) = 1]\right| \leq negl(n)$$

Syntax

- $Gen(1^n)$: Outputs public key and secret key pair (pk, sk).
- $Sign_{sk}(m)$: Outputs a signature σ on the message m.
- $Vrfy_{pk}(m, \sigma)$: Outputs $0/1$.

Correctness: For all n, except for negligible choices of (pk, sk), it holds that for all m, $Vrfy_{pk}\big(m, Sign_{sk}(m)\big) = 1$.

Unforgeability/Security of Digital Signature

$\mathrm{Forge}_{A,\Pi}(1^n)$

1. Sample *(pk,sk)* $\leftarrow$ Gen(1^n).
2. Let (m^*, σ^*) be the output of $A^{Sign_{sk}(\cdot)}(pk)$. Let M be the list of queries A makes.
3. Output 1 if $Vrfy_{pk}(m^*, \sigma^*) = 1 \wedge m^* \notin M$ and 0 otherwise.

$\Pi = (Gen, Sign, Vrfy)$ is existentially unforgeable under adaptive chosen attack if

$\forall$ PPT A it holds that:

$\Pr\big[\mathrm{Forge}_{A,\Pi} = 1\big] \leq$ negl(n)

Identity-Based Encryption (IBE) [Shamir84]

Four Algorithms: (S, K, E, D)

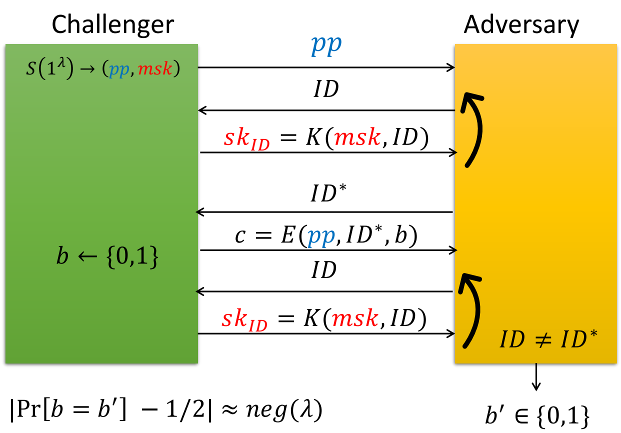
$S(1^\lambda) \quad \rightarrow (pp, msk)$ — pp are public parameters, msk is the master secret-key

$K(msk, ID) \quad \rightarrow sk_{ID}$ — sk_{ID} secret key for ID

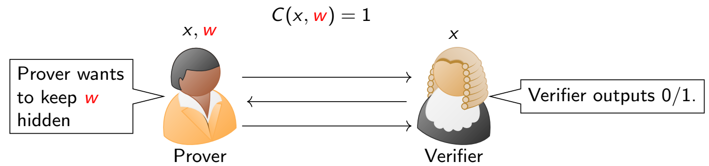
$E(pp, ID, m) \rightarrow c$ — encrypt using pp and ID

$D(sk_{ID}, c) \quad \rightarrow m$ — decrypt c using sk_{ID}

Security of IBE [BF01]



Zero-Knowledge Proof System



- Syntax: Two algorithms, $P(1^n, x, w)$ and $V(1^n, x)$.
- Completeness: Honest prover convinces an honest verifier with *overwhelming* probability.

$$\Pr[V \text{ outputs 1 in the interaction } P(1^n, x, w) \leftrightarrow V(1^n, x)] = 1 - \mathrm{neg}(n)$$

- Soundness: A PPT cheating prover P^* cannot make a Verifier accept a false statement. For all PPT P^*, x such that $\forall w, C(x, w) = 0$then we have that

$$\Pr[V \text{ outputs 1 in the interaction } P^*(1^n, x) \leftrightarrow V(1^n, x)] = \mathrm{neg}(n)$$

- Zero-Knowledge: The proof doesn't leak any information about the witness w. $\exists$ a PPT simulator $\mathcal{S}$ that for all PPT V^*, x, w such that $C(x, w) = 1$, we have that $\forall$ PPT D:

$$\left| \Pr[D(V^*\text{'s view in } P(1^n, x, w) \leftrightarrow V^*(1^n, x)) = 1] - \Pr[D(\mathcal{S}^{V^*}(1^n, x)) = 1] \right| \leq \mathrm{neg}(n)$$